

C# Programlama

Koleksiyonlar ve Genel Sınıflar

Emir Öztürk

Genel Türler

- Generic
- Birden fazla tür için tanımlanabilir sınıflar
- <T>
- Yeniden kullanılabilirlik
- En çok görülen kullanım alanı
 - Koleksiyonlar
 - Akışlar

Genel Türler

```
namespace Uygulama
{
    2 references
    class Sinif
    {
        1 reference
        internal void Yazdir(string v)
        {
            Console.WriteLine(v);
        }

        1 reference
        internal void Yazdir(int v)
        {
            Console.WriteLine(v);
        }
    }
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Sinif s = new Sinif();
            s.Yazdir("String");
            s.Yazdir(1);
        }
    }
}
```



```
namespace Uygulama
{
    2 references
    class Sinif
    {
        2 references
        internal void Yazdir<T>(T v)
        {
            Console.WriteLine(v);
        }
    }
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Sinif s = new Sinif();
            s.Yazdir("String");
            s.Yazdir(1);
        }
    }
}
```

Genel Türler

```
namespace Uygulama
{
    2 references
    class Sinif
    {
        1 reference
        internal void Yazdir(string v)
        {
            Console.WriteLine(v);
        }

        1 reference
        internal void Yazdir(int v)
        {
            Console.WriteLine(v);
        }
    }
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Sinif s = new Sinif();
            s.Yazdir("String");
            s.Yazdir(1);
        }
    }
}
```



```
5 references
class Sinif<T>
{
    T degisken;
    2 references
    public Sinif(T degisken)
    {
        this.degisken = degisken;
    }
    2 references
    internal void Yazdir()
    {
        Console.WriteLine(degisken);
    }
}
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Sinif<string> s = new Sinif<string>("String");
        s.Yazdir();
        Sinif<int> s2 = new Sinif<int>(3);
        s.Yazdir();
    }
}
```

Genel Türler

- T bir değişken olarak veya dizi olarak kullanılabilir.

```
4 references
class Sinif<T>
{
    T degisken;
    T[] dizi;
    2 references
    internal void Yazdir()
    {
        for (int i = 0; i < dizi.Length; i++)
            Console.Write(dizi[i]);
    }
}
```

Genel Türler

- T üzerinde aritmetik işlemler gerçekleştirilemez.
- Bunun için ek metotlar yazılması gerekmektedir.

```
class Sinif<T>
{
    T degisken;
    T[] dizi;
    2 references
    internal void Yazdir()
    {
        T sonuc;
        for (int i = 0; i < dizi.Length; i++)
            sonuc += dizi[i];
    }
}
```

CS0019: Operator '+=' cannot be applied to operands of type 'T' and 'T'

```
class Sinif<T>{
    T[] dizi;
    0 references
    internal void Hesapla(){
        dynamic sonuc=0;
        for (int i = 0; i < dizi.Length; i++)
            sonuc = topla(sonuc, dizi[i]);
    }
    1 reference
    private T topla(T sayi1, T sayi2){
        dynamic a = sayi1;
        dynamic b = sayi2;
        return a + b;
    }
}
```

Koleksiyonlar

- Diziler
 - Statik
 - Daha iyi performans
- Koleksiyonlar
 - Dinamik
 - Farklı ihtiyaçlara çözümler
 - Sınıf - Örnek alınmalı

Koleksiyonlar

- List<T>
- Dictionary<TKey,TValue>
- HashSet<T>
- Queue<T>
- SortedList<T>
- Stack<T>

Koleksiyonlar

List

- Dinamik
- Dizi
- Tek türden elemanların listesi

Koleksiyonlar

List

- Temel tip listeleri oluşturulabilir.

```
List<string> stringListesi = new List<string>();  
List<int> intListesi = new List<int>();
```

- Sınıf listeleri oluşturulabilir.

```
List<Sinif> sinifListesi = new List<Sinif>();  
List<FileStream> dosyaListesi = new List<FileStream>();
```

- Sınıf alabildiği için List listeleri oluşturulabilir.

```
var listListesi = new List<List<string>>();  
var listListesiListesi = new List<List<List<string>>>();
```

Koleksiyonlar

List

- Add()
- AddRange()
- Remove()
- RemoveAt()
- Count
- ToArray()
- Clear()

```
static void Main(string[] args)
{
    List<string> stringListesi = new List<string>();
    List<string> altListe = new List<string>() { "deger1", "deger2", "deger3" };
    stringListesi.Add("a");
    stringListesi.AddRange(altListe);
    stringListesi.Remove("deger1");
    stringListesi.RemoveAt(2);
    int uzunluk = stringListesi.Count;
    string[] dizi = stringListesi.ToArray();
    stringListesi.Clear();
}
```

Koleksiyonlar

Dictionary

- Anahtar ve değer alanları

```
int deger=0;  
Dictionary<string, int> degerler = new Dictionary<string, int>();
```

- Her iki alan da <T>

```
var degerler1 = new Dictionary<string, int>();  
var degerler2 = new Dictionary<int, string>();  
var degerler3 = new Dictionary<int, float>();  
var degerler4 = new Dictionary<string, Sinif>();
```

- Anahtar ile değere erişim

```
int deger=0;  
Dictionary<string, int> degerler = new Dictionary<string, int>();  
degerler["anahtar"] = deger;  
deger = degerler["anahtar"];
```

Koleksiyonlar

Dictionary

- Add()
- Count
- Keys()
- Values()
- ContainsKey()
- ContainsValue()

```
Dictionary<string, int> degerler = new Dictionary<string, int>();
degerler.Add("Anahtar", 1);
Console.WriteLine(degerler.Count);
foreach (var key in degerler.Keys)
    Console.WriteLine(key);
foreach (var value in degerler.Values)
    Console.WriteLine(value);
Console.WriteLine(degerler.ContainsKey("Anahtar"));
Console.WriteLine(degerler.ContainsValue(1));
```

Koleksiyonlar

Hashset

- Tek anahtar alanı <T>

```
HashSet<int> set = new HashSet<int>();
```

- Her eleman tekil

```
bool sonuc = set.Add(1);  
sonuc = set.Add(1);
```

- Elemanların varlığının kontrolü

```
bool varMi = set.Contains(1);
```

- Eleman silme

```
set.Remove(1);  
set.RemoveWhere(x => x > 5);
```

Koleksiyonlar

Sortedlist

- Sıralama ölçütü ve sıralanacak değer şeklinde iki alan

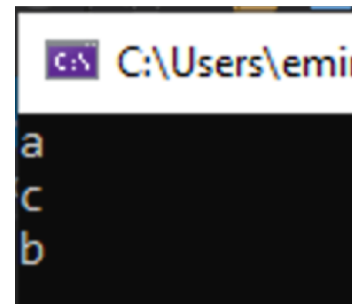
```
SortedList<int, string> siraliListe = new SortedList<int, string>();
```

- Değer eklemek için add metodu

```
siraliListe.Add(1, "a");  
siraliListe.Add(3, "b");  
siraliListe.Add(2, "c");
```

- Sıralanmış değerleri almak için values özelliği

```
List<string> degerler = siraliListe.Values.ToList();  
foreach (var deger in degerler)  
    Console.WriteLine(deger);
```



```
C:\Users\emir  
a  
c  
b
```

Koleksiyonlar

Queue

- Kuyruk implementasyonu (ilk giren ilk çıkar fifo)

```
Queue<string> kuyruk = new Queue<string>();
```

- Kuyruğa ekleme

```
kuyruk.Enqueue("Eleman");
```

- Kuyruktan eleman alma

```
string alinan = kuyruk.Dequeue();
```

- Sıradaki elemana kuyruktan çıkarmadan bakma

```
string siradaki = kuyruk.Peek();
```

- Eleman sayısı

```
int sayi = kuyruk.Count;
```


Koleksiyonlar

Stack

- Lifo
- <T>
- Kuyruk metotları ile aynı

```
Stack<string> stack = new Stack<string>();  
stack.Push("Eleman");  
string alinan = stack.Pop();  
int sayi = stack.Count;  
string siradaki = stack.Peek();
```